**Objectives**

After studying this unit, you will be able to:

* Describe the characteristics of multiprocessor
* Discuss the uses of multiprocessors
* Explain interconnection structures
* Explain interprocessor communication and synchronization

## Introduction

Multiprocessor is a single computer that has multiple processors. It is possible that the processors in the multiprocessor system can communicate and cooperate at various levels of solving a given problem. The communications between the processors take place by sending messages from one processor to another, or by sharing a common memory.

Both multiprocessors and multicomputer systems share the same fundamental goal, which is to perform the concurrent operations in the system. However, there is a significant difference between multicomputer systems and multiprocessors. The difference exists depending on the extent of resource sharing and cooperation in solving a problem. A multicomputer system includes numerous autonomous computers which may or may not communicate with each other. However, a single operating system that provides communication between processors and their programs on the process, data set, and data element level, controls a multiprocessor system.

**Notes** The interprocessor communication is carried out with the help of shared memories or through an interrupt network. Most significantly, a single operating system that provides interactions between processors and their programs at different levels, control the whole system.



*Did u know?* Processors share access to general sets of memory modules, Input/Output channels, and also peripheral devices. All processors have their individual local memory and Input/ Output devices along with shared memory.

## Multiprocessors

Multiprocessor is a data processing system that can execute more than one program or more than one arithmetic operation simultaneously. It is also known as multiprocessing system. Multiprocessor uses with more than one processor and is similar to multiprogramming that allows multiple threads to be used for a single procedure. The term ‘multiprocessor’ can also be used to describe several separate computers running together. It is also referred to as clustering. A system is called multiprocessor system only if it includes two or more elements that can implement instructions independently. A multiprocessor system employs a distributed approach. In distributed approach, a single processor does not perform a complete task. Instead more than one processor is used to do the subtasks.

Some of the major characteristics of multiprocessors include:

1. ***Parallel Computing***: This involves simultaneous application of multiple processors. These processors are developed using a single architecture in order to execute a common task. In general, processors are identical and they work together in such a way that the users are under the impression that they are the only users of the system. In reality, however, there are many users accessing the system at a given time.
2. ***Distributed Computing:*** This involves the usage of a network of processors. Each processor in this network can be considered as a computer in its own right and have the capability to solve a problem. These processors are heterogeneous, and generally one task is allocated to a single processor.
3. ***Supercomputing:*** This involves usage of the fastest machines to resolve big and computationally complex problems. In the past, supercomputing machines were vector computers but at present, vector or parallel computing is accepted by most of the people.
4. ***Pipelining:*** This is a method wherein a specific task is divided into several subtasks that must be performed in a sequence. The functional units help in performing each subtask. The units are attached in a serial fashion and all the units work simultaneously.
5. ***Vector Computing:*** It involves usage of vector processors, wherein operations such as ‘multiplication’ is divided into many steps and is then applied to a stream of operands (“vectors”).
6. ***Systolic:*** This is similar to pipelining, but units are not arranged in a linear order. The steps in systolic are normally small and more in number and performed in a lockstep manner. This is more frequently applied in special-purpose hardware such as image or signal processors.

A multiprocessor system has the following advantages:

1. It helps to improve the cost or performance ratio of the system.
2. It helps to fit the needs of an application, when several processors are combined. At the same time, a multiprocessor system avoids the expenses of the unnecessary capabilities of a centralized system. However, this system provides room for expansion.
3. It helps to divides the tasks among the modules. If failure happens, it is simple and cheaper to identify and replace the malfunctioning processor, instead of replacing the failing part of complex processor.
4. It helps to improve the reliability of the system. A failure that occurs in any one part of a multiprocessor system has a limited effect on the rest of the system. If error occurs in one processor, a second processor may take up the responsibility of doing the task of the processor in which the error has occurred. This helps in enhancing the reliability of the system at the cost of some loss in the efficiency.

#### 13.1.1 Coupling of Processors

There are two types of multiprocessor systems and they are:

1. ***Tightly-coupled Multiprocessor System***: This system has many CPUs that are attached at the bus level. Tasks and/or processors interact in a highly synchronized manner. The CPUs have access to a central shared memory and communicate through a common shared memory.
2. ***Loosely-coupled Multiprocessor System***: This multiprocessor system is often referred to as clusters. These systems operate based on single or dual processor commodity computers interconnected through a high speed communication system. Tasks or processors do not communicate in a synchronized manner as done in tightly-coupled multiprocessor systems. They communicate through message passing packets. This system has a high overhead for data exchange and uses distributed memory system.

 *Example:* The best example for a loosely-coupled multiprocessor system is a Linux Beowulf cluster. The example for a tightly-coupled multiprocessor system is mainframe system.

##### Granularity of Parallelism

When you talk about parallelism, you need to know the concept of granularity. The granularity of parallelism specifies the size of the computations that are carried out at the same time between synchronizations. Granularity is referred to as the level to which a system is divided into small parts, either the system itself or its explanation or observation. Granularity of parallelism is of three types. They are:

1. ***Coarse-grain***: A task is divided into a handful of pieces, where each piece is performed with the help of a powerful processor. Processors are heterogeneous. Communication/computation ratio is very high.
2. ***Medium-grain***: A task is divided into tens to few thousands of subtasks. Processors here usually run the same code. Computation ratio is more often hundreds or more.
3. ***Fine-grain:*** A task is divided into thousands to millions of small subtasks that are implemented using very small and simple processors, or through pipelines. Processors have instructions broadcasted to them. The computation ratio is more often 1 or less.

##### Memory

We are aware of the concepts of memory and the memory hierarchy. The different categories of memory discussed in the previous units are main memory, cache memory, and virtual memory. In this section, the different types of memory are listed. They are:

1. Shared (Global) Memory:
   1. All processors can access a global memory space.
   2. Processors can also have some local memory.

**Notes**

**Notes** Figure 13.1 depicts shared memory.

|  |  |  |
| --- | --- | --- |
|  | **Figure 13.1: Shared Memory** |  |
|  |  |
|  | | |

1. Distributed (Local, message-passing) Memory:
   1. All the memory units are associated with the processors.
   2. A message must be sent to another processor’s memory to retrieve information from that memory.

Figure 13.2 depicts distributed memory.

|  |  |  |
| --- | --- | --- |
|  | **Figure 13.2: Distributed Memory** |  |
|  |  |
|  | | |

1. ***Uniform Memory:*** Every processor takes the same time to reach all memory locations.
2. ***Non-uniform Memory Access:*** Memory access is not uniform. It is in contrast to the uniform memory.

##### Shared Memory Multiprocessors

In shared-memory multiprocessors, there are numerous processors accessing one or more shared memory modules. The processors may be physically connected to the memory modules in many ways, but logically every processor is connected to every memory module.

One of the major characteristics of shared memory multiprocessors is that all processors have equally direct access to one large memory address space.

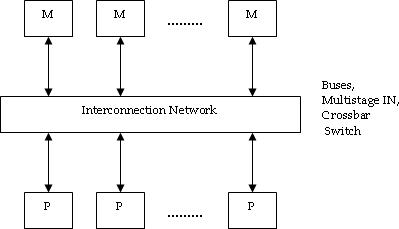
The limitation of shared memory multiprocessors is memory access latency.

 *Example:* Bus and cache-based systems: Encore Multimax, Sequent BalanceMultistage IN- based systems: Ultracomputer, Butterfly, RP3, HEPCrossbar switch-based systems: C mmp, Alliant FX/8

Figure 13.3 depicts shared memory multiprocessors.

###### Notes

**Figure 13.3: Shared Memory Multiprocessors**

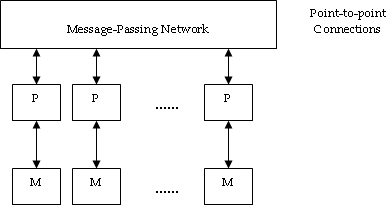
Shared memory multiprocessors have a major benefit over other multiprocessors, because all the processors share the same view of the memory.

These processors are also termed as Uniform Memory Access (UMA) systems. This term denotes that memory is equally accessible to every processor, providing the access at the same performance rate.

##### Message-Passing Multiprocessors

In a message-passing multiprocessor system, a method for conveying messages between nodes, and a node and a method, in order to format the same in a message-passing computer system is specified. Network interface is an example of the message-passing multiprocessor system. In the network interface for a computer system, there exists:

1. Multiple nodes linked with one another through an interconnection network for communication of messages.
2. More than one processor and a local shared memory that are linked with one another through a node bus.

**Notes** Figure 13.4 depicts message-passing multiprocessors.

**Figure 13.4: Message-passing Multiprocessors**

Some of the important characteristics of message-passing multiprocessors are:

* 1. Computers are interconnected.
  2. All processors have their own memory and they communicate through message -passing.

 *Example:* Tree structure: Teradata, DADO

Mesh-connected: Rediflow, Series 2010, J-MachineHypercube: Cosmic Cube, iPSC, NCUBE, FPS T Series, Mark III

Limitations of message-passing multiprocessors are communication overhead and difficulty in programming.

## Uses of Multiprocessors

Use of multiprocessor systems in real-time applications is becoming popular. One of the major reasons for this popularity is the recent drop in the cost of these systems. At present, dual processor machines are available at fifty to sixty thousand rupees, and it is predicted that the prices are going to drop even further. The faster response time and fault-tolerance feature of such systems are the other reasons that attract real-time system developers to install multiprocessor systems.

It is to be noted that using a multiprocessor is more beneficial than using independent processors. The parallelism existing within each multiprocessor helps in gaining localized high performance and also maintains extensive multithreading for the fine-grained parallel programming models. The thread block has individual threads that execute together within a multiprocessor to allocate data.

For maintaining area and power efficiency, the multiprocessor shares large and complex units among the different processor cores, which also include the instruction cache, the multithreaded instruction unit, and the shared memory RAM.

One of the main advantages of multiprocessor is shared memory programming model. Shared- memory multiprocessors have a major advantage over other multiprocessors, as all the other processors share the same view of the memory. These processors are also termed as Uniform Memory Access (UMA) systems. This term indicates that all processors can equally access the memory with the same performance.

The popularity of the shared-memory systems is just not due to the demand for high performance computing. These systems also provide high throughput for a multiprocessing load. They also work efficiently as high-performance database servers, Internet servers, and network servers. As more processors are added, the throughput of these systems is increased linearly.

Multiprocessors also find their applications in various domains which include:

1. ***Server Workload:*** This includes many concurrent updates, lookups, searches, queries, and so on. Processors deal with different requests.

 *Example:* Database for airline reservation

1. ***Media Workload:*** Processors compress/decompress different parts of image/frames. This includes compressing/decompressing of different parts of image/frames.
2. ***Scientific Computing:*** This includes large grids that integrate changes over time, and each processor computes for a part of the grid.

 *Example:* Protein folding, aerodynamics, and weather simulation.

## Interconnection Structures

The structures that are used to connect the memories and processors (and between memories and I/O channels if required), are called interconnection structures. A multiprocessor system is formed by elements such as CPUs, peripherals, and a memory unit that is divided into numerous separate modules. There can exist different physical configurations for the interconnection between the elements. The physical configurations are based on the number of transfer paths existing between the processors and memory in a shared memory system or among the processing elements in a loosely coupled system. An interconnection network is established using several physical forms available. Some of the physical forms include:

1. Time-shared common bus
2. Multiport memory
3. Crossbar switch
4. Multistage switching network
5. Hypercube system

##### Operation of Bus

Bus is defined as a group of signal lines that carry module-to-module communication. Here, data highways connect several digital system elements. Each processor (and memory) is connected to a common bus. Memory access is moderately uniform, but it is less scalable.

###### Notes

**Notes** Figure 13.5 depicts operation of bus.

|  |  |  |
| --- | --- | --- |
|  | **Figure 13. 5: Operation of Bus** |  |
|  |  |
|  | | |

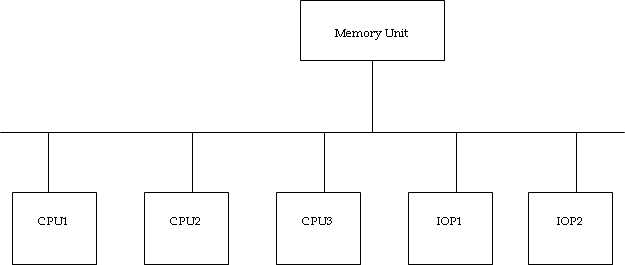
In figure 13.5:

Master Device (M2, M3, M4): This is a device that initiates and controls the communication. Slave Device (S5, S6, S8): This is a responding device.

As depicted in figure 13.5, if M2 wishes to communicate with S6,

* 1. M2 sends signals (address) on the bus that causes S6 to respond.
  2. M2 sends data to S6, or S6 sends data to M2. (determined by the command line)
     1. **Time-shared Common Bus**

In time-shared common bus, there are numerous processors connected through a common path to the memory unit in a common-bus multiprocessor system. Figure 13.6 shows organization of time-shared common bus for five processors. At any specified time, only one processor can communicate with the memory or another processor. The processor that is in control of the bus at the time performs transfer operations. Any processor that wants to initiate a transfer must first verify the availability status of the bus.

Once the bus is available, the processor can establish a connection with the destination unit to initiate the transfer. A command is issued to inform the destination unit about the function to be performed. The receiving unit identifies its address in the bus, and then responds to the control signals from the sender, after which the transfer is initiated. As all processors share a common bus, it is possible that the system may display some transfer conflicts. Incorporation of a bus controller that creates priorities among the requesting units helps in resolving the transfer conflicts.

**Figure 13.6: Organization of a Time-Shared Common Bus**

There is a restriction of one transfer at a time for a single common-bus system. This means that other processors are busy with internal operations or remain idle waiting for the bus when one processor is communicating with the memory. Hence, the speed of the single path limits the total overall transfer rate within the system. The system processors are kept busy through the execution of two or more independent buses, to allow multiple bus transfers simultaneously. However, this leads to increase in the system cost and complexity.

Figure 13.7 depicts a more economical execution of a dual bus structure for multiprocessors.

|  |  |  |
| --- | --- | --- |
|  | **Figure 13.7: System Bus Structure for Multiprocessors** |  |
|  |  |
|  | | |

In figure 13.7 we see that there are many local buses, and each bus is connected to its own local memory, and to one or more processors. Each local bus is connected to a peripheral, a CPU, or any combination of processors. Each local bus is linked to a common system bus using a system bus controller.

The I/O devices connected to both the local I/O peripherals and the local memory is available to the local processor. All processors share the memory connected to the common system bus. When an IOP is connected directly to the system bus, the Input/Output devices attached to it are made available to all processors. At any specified time, only one processor can communicate with the shared memory, and other common resources through the system bus. All the other processors are busy communicating with their local memory and I/O devices.

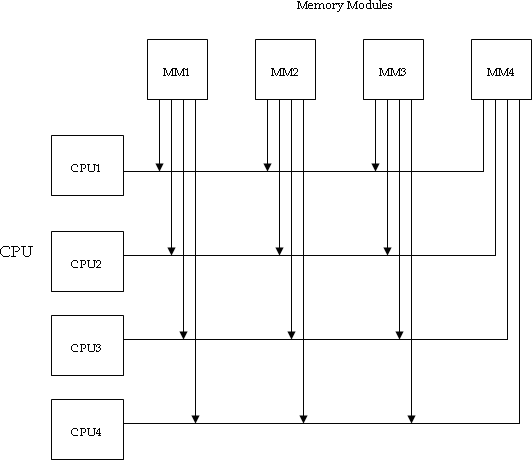
#### Multiport Memory

Multiport memory is a memory that helps in providing more than one access port to separate processors or to separate parts of one processor. A bus can be used to achieve this kind of an access. This mechanism is applicable to interconnected computers too. A multiport memory system uses separate buses between each CPU and each memory module. Figure 9.8 depicts a multiport memory system for four CPUs and four Memory Modules (MMs). Every processor bus is connected to each memory module. A processor bus consist three elements; namely: address, data, and control lines. These elements are needed to communicate with memory. Memory module has four ports and each port contains one of the buses. It is necessary for a module to have internal control logic to verify which port will have access to memory at any specified time. Assigning fixed priorities to each memory port helps in resolving memory access conflicts. The priority for memory access

###### Notes

**Notes** related to each processor is created with the physical port position that its bus occupies in each module. Consequently, CPU1 has priority over CPU2, CPU2 has priority over CPU3, and CPU4 has the least priority.

**Figure 13.8: Multiport Memory Organization**

The multiport memory organization has an advantage of high transfer rate. This is because of several paths between memory and processors. The only drawback is that it needs expensive memory control logic and more number of cables and connectors. Therefore, this interconnection structure is usually suitable for systems having small number of processors.

#### Crossbar Switch

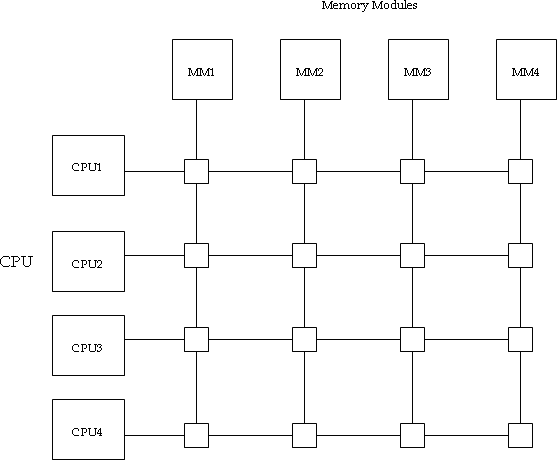
In a network, a device that helps in channeling data between any two devices that are connected to it, up to its highest number of ports is a crossbar switch. The paths set up between devices can be fixed for some period of time or changed when wanted.

In a crossbar switch organization, there are several cross points that are kept at intersections between processor buses and memory module paths.

Figure 13.9 shows a crossbar switch interconnection between four memory modules and four CPUs.

###### Notes

**Figure 13.9: Crossbar Switch**

In figure 13.9, the small square in each crosspoint indicates a switch. This switch determines the path starting from a processor to a memory module. There is control logic for each switch point to set up the transfer path between a memory module and a processor. It checks the address that is placed in the bus to verify if its particular module is addressed. It also allows resolving multiple requests to get access to the same memory module on a predetermined priority basis.

**Notes** The functional design of a crossbar switch connected to one memory module is depicted in figure 13.10.

|  |  |  |
| --- | --- | --- |
|  | **Figure 13.10: Block Diagram of Crossbar Switch** |  |
|  |  |
|  | | |

The circuit includes multiplexers that choose the data, address, and control from one CPU for communication with the memory module. The arbitration logic establishes priority levels to choose one CPU when two or more CPUs try to get access to the same memory. The binary code controls the multiplexers. A priority encoder generates this binary code within the arbitration logic.

*Notes* A crossbar switch organization maintains and supports simultaneous transfers from memory modules, since there is a separate path related with each module. On the other hand, the hardware necessary to implement the switch may be quite large and complex.

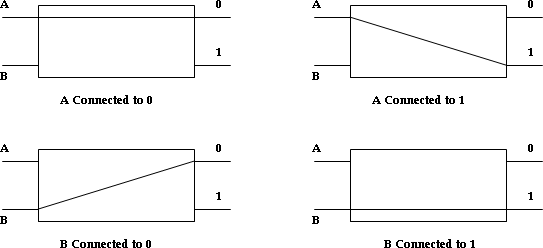
#### Multistage Switching Network

The network that is built from small (for example, 2 x 2 crossbar) switch nodes along with a regular interconnection pattern is a multistage switching network. Two-input, two-output interchange switch is a fundamental element of a multistage network. There are two inputs marked A and B, and two outputs marked 0 and 1 in the 2 x 2 switch as shown in figure 13.11.

Figure 13.11 depicts operation of a 2x2 interchange switch.

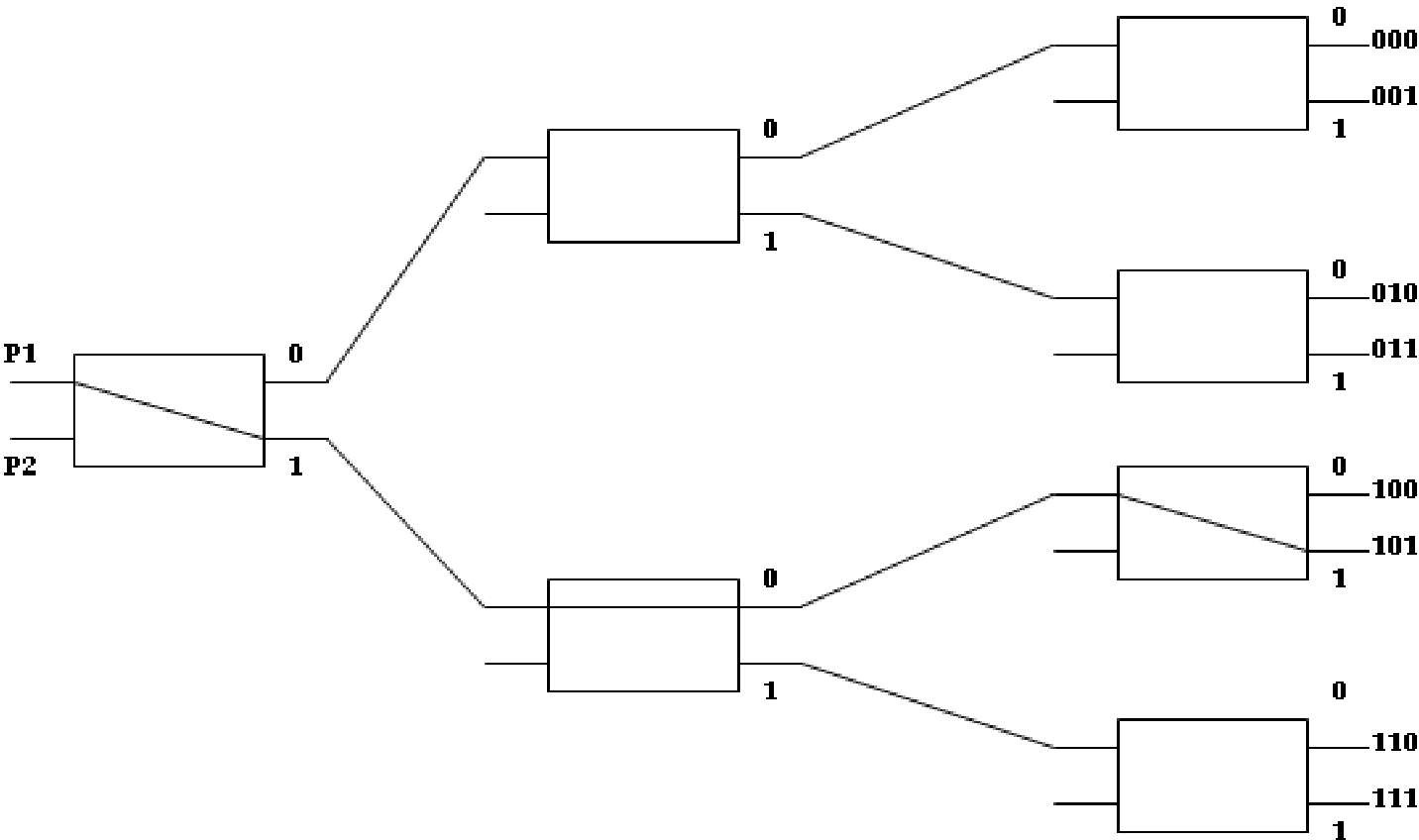
###### Notes

**Figure 13.11: Operation of a 2 x 2 Interchange Switch**

As depicted in figure 13.11, there are control signals associated with the switch. The control signals establish interconnection between the input and output terminals. The switch can connect input A to either of the outputs. Terminal B of the switch acts in a same way. The switch can also arbitrate between conflicting requests. In case, inputs A and B request the same output terminals, it is possible that only one of the inputs is connected and the other is blocked.

It is possible to establish a multistage network to control the communication between numerous sources and destinations. The multistage network is established with the help of 2 x 2 switch as a building block.

**Notes** Consider the binary tree shown in figure 13.12 to see how this is carried out.



**Figure 13.12: Binary Tree with 2 x 2 Switches**

The two processors P1 and P2 are linked through switches to eight memory modules labeled in binary, starting from 000 through 111. The path starting from source to destination is determined from the binary bits of destination number. The first bit of the destination number helps in indicating the first level’s switch output. The second bit identifies the second level’s switch output, and the third bit specifies the third level’s switch output.

 *Example:* As shown in figure 13.12, in order to make a connection between P1 and memory 101, it is important to create a path from P1 to output 1 in the third-level switch, output 0 in the second-level switch, and output 1 in the third-level switch. Hence, it is evident that either P1 or P2 must be connected to any one of the eight memories.

It is also evident that certain request patterns however cannot be satisfied simultaneously.

 *Example:* As shown in figure 13.12, if P1 is connected to one of the destinations 000 through 011, then it is possible to connect P2 to only one of the destinations 100 through 111.

There are many topologies for multistage switching networks that help to:

1. Control the processor-memory communication in a tightly-coupled multiprocessor system.
2. Control the communication between the processing components in a loosely-coupled system.

Omega switching network is one such topology that is depicted in the figure 13.13. There exists exactly one path from source to any specific destination in this configuration. However, certain request patterns cannot be connected simultaneously. For example, it is not possible to connect any two sources simultaneously to destinations 000 and 001.

Figure 13.13 depicts 8x8 Omega switching network.

|  |  |  |
| --- | --- | --- |
|  | **Figure 13.13: 8 x 8 Omega Switching Network** |  |
|  |  |
|  | | |

As depicted in figure 13.13, a specific request is started in the switching network through the source that sends a 3-bit pattern depicting the destination number. Every level checks a different bit to determine the 2 x 2 switch setting as the binary pattern moves through the network. Level 1 examines the most important bit, level 2 examines the middle bit, and level 3 examines the least important bit. When the request appears on input 2 x 2 switch, it is routed to the lower output if the specified bit is 1 or to the upper output if the specified bit is 0.

The source is considered to be a processor and the destination is considered as a memory module in a tightly-coupled multiprocessor system. The path is set when the first pass is through the network. If the request is read or write the address is transferred into memory, and then the data is transferred in either direction using the succeeding passes. Both the destination and the source are considered to be processing elements in a loosely-coupled multiprocessor system. The source processor transfers a message to the destination processor once the path is established.

#### Hypercube Interconnection

The hybercube is considered to be a loosely coupled system. This system is composed of N = 2n processors that are interconnected in an n-dimensional binary cube. Each processor indicates a node of the cube. Although it is expected to refer to every node as having a processor, in effect it not only has a CPU but also local memory and I/O interface. Every processor contains direct communication paths to n other neighbor processors. These paths relates to the edges of the cube. The processors can be assigned with 2n distinct n-bit binary addresses. Each processor address differs from that of each of its n neighbors by exactly one bit position.

*Did u know?* The hypercube interconnection is also referred to as binary *n*-cube multiprocessor structure.

###### Notes

**Notes** Figure 13.14 depicts the hypercube structure for n, wherein n = 1, 2, and 3.

|  |  |  |
| --- | --- | --- |
|  | **Figure 13.14: Hypercube Structures for n = 1, 2, and 3** |  |
|  |  |
|  | | |

A depicted in figure 13.14, a one-cube structure contains n = 1 and 2n = 2. It has two processors that are interconnected by a single path. A two-cube structure contains n = 2 and 2n = 4. It has four nodes that are interconnected as a square. There are eight nodes interconnected as a cube in a three-cube structure. There are 2n nodes in an n-cube structure with a processor existing in every node.

A binary address is assigned to every node such that the addresses of two neighbors vary in exactly one bit position.

 *Example:* As shown in figure 13.13, the three neighbors of the node having address 100 in a three-cube structure are 000, 110, and 101. Each of these binary numbers vary from address 100 by one bit value.

Routing messages through an n-cube structure may require one to n links, starting from a source node to a destination node.

 *Example:* As shown in figure 13.12, it is possible for node 000 to communicate directly with node 001 in a three-cube structure. To communicate from node 000 to node 111, the message has to travel through at least three links.

Computing the exclusive-OR of the source node address with the destination node address helps in developing a routing procedure. The resulting binary value has 1 bit relating to the axes on which the two nodes vary. Later, the message is sent along any one of the axes.

 *Example:* A message at 010 being sent to 001 generates an exclusive-OR of the two addresses equivalent to 011 in a three-cube structure. It is possible to send the message along the second axis to 000 and then through the third axis 001.

###### Notes

*Example:* The Intel iPSC complex is considered to be a representative of the hypercube architecture.

The Intel iPSC has 128 (*n* = 7) microcomputers connected through communication channels. Each node has a CPU, local memory, floating-point processor, and serial communication interface units. The individual nodes work independently on data saved in local memory according to the resident programs. It is evident that the programs and data at every node is received through a message- passing system from other nodes or from a cube manager. Application programs are developed and gathered on the cube manager and then downloaded to the individual nodes. Computations are allocated through the system and implemented concurrently.



*Task* Visit <http://ed-thelen.org/comp-hist/intel-iPSC-860.html> and discuss about the hypercube architecture and special features of Intel iPSC/860.

## Interprocessor Communication and Synchronization

A multiprocessor system has various processors that must be provided with a facility to communicate with each other. Using a common I/O channel, a communication path is established. The most frequently used procedure in a shared memory multiprocessor system is to set aside a part of the memory that is available to all processors. The major use of the common memory is to work as a message center similar to a mailbox, where every processor can leave messages for other processors and pick up messages meant for it.

The sending processor prepares a request, a message, or a procedure, and then places it in the memory mailbox. The receiving processor can check the mailbox periodically to determine if there are valid messages in it, as a processor identifies a request only while polling messages. However, the response time of this procedure may be time consuming. The sending processor has a more efficient procedure, and the procedure involves alerting the receiving processor directly using an interrupt signal. This procedure is achieved with the help of software initiated interprocessor interrupt initialized in one processor, which when implemented generates an external interrupt condition in a second processor. This interrupt informs the second processor that processor one has inserted a new message in its mailbox.

*Notes* Status bits present in common memory are usually used to determine the condition of the mailbox, if it has meaningful data, and for which processor it is intended.

A multiprocessor system has other shared resources in addition to shared memory.

 *Example:* An IOP to which a magnetic disk storage unit is connected, is available to all CPUs. This helps in providing a facility for sharing of system programs stored in the disk.

A communication path can be established between two CPUs through a link between two IOPs, which connects two different CPUs. This kind of link allows each CPU to treat the other as an I/O device, such that messages can be transferred through the I/O path.

There should be a provision for assigning resources to processors to avoid inconsistent use of shared resources by many processors. This job is given to the operating system. The three organizations that are used in the design of operating system for multiprocessors include:

1. Master-slave configuration
2. Separate operating system
3. Distributed operating system

In a master-slave configuration mode, one processor, designated the master, always implements the operating system functions. The remaining processors, designated as slaves, do not execute

**Notes** operating system functions. If a slave processor requires an operating system service, then it should request it by interrupting the master.

Each processor can implement the operating system routines that it requires in the separate operating system organization. This kind of organization is more appropriate for loosely-coupled systems wherein, every processor needs to have its own copy of the entire operating system.

The operating system routines are shared among the available processors in the distributed operating system organization. However, each operating system function is allocated to only one processor at a time. This kind of organization is also termed as a floating operating system because the routines float from one processor to another, and the implementation of the routines are allocated to different processors at different times.

The memory is distributed among the processors and there is no shared memory for sending information in a loosely-coupled multiprocessor system. Message passing system through I/O channels is used for communication between processors. The communication is started by one processor calling a procedure that exists in the memory of the processor with which it has to communicate. A communication of channel is established when both the sending processor and the receiving processor recognize each other as source and destination. A message is then sent to the nodes with a header and different data objects that are required for communication between the nodes. In order to send the message between any two nodes, several possible paths are available. The operating system of each node has the routing information which indicates the available paths to send a message to different nodes.

The communication efficiency of the interprocessor network depends on four major factors and they are:

* 1. Communication routing protocol
  2. Processor speed
  3. Data link speed
  4. Topology of the network

##### Interprocessor Synchronization

Synchronization is a communication of control information between processors. Synchronization helps to:

1. Implement the exact sequence of processes.
2. Ensure mutually exclusive access to allocated writable data.

Synchronization refers to a special case where the control information is the data employed to communicate between processors. Synchronization is necessary to implement the exact sequence of processes and to ensure mutually exclusive access to shared writable data.

There are many mechanisms in multiprocessor systems to handle the synchronization of resources. The hardware directly implements low-level primitives. These primitives act as essential mechanisms that enforce mutual exclusion for more difficult mechanisms executed in software. Many hardware mechanisms for mutual exclusion are developed. However, the use of a binary semaphore is considered to be one of the most popular mechanisms.

The following are the methods to achieve synchronization.

Synchronization can be achieved by mutual exclusion with a semaphore. Semaphores are considered to be the means of addressing the requirements of both task synchronization and mutual exclusion. Mutual exclusion includes a processor to eliminate or lock out access to allocated resource by other processors when it is in a **Critical Section.**

##### Mutual Exclusion with a Semaphore

Appropriately operating multiprocessor system must provide a mechanism that would ensure systematic access to shared memory and other shared resources. This is required to protect data, since two or more processors can change the data simultaneously. This mechanism is referred to as mutual exclusion. A multiprocessor system must have mutual exclusion to allow one processor to rule out or lock out access to an allocated resource by other processors when it is in a critical section. A critical section is defined as a program sequence which once started must complete implementation before another processor accesses the same allocated resource.

###### Notes

*Notes* A semaphore is considered to be a software-controlled flag stored in a memory location such that all processors can access.

When the semaphore is set to one, it indicates that a processor is implementing a critical program, and the shared memory is unavailable to other processors. When the semaphore is set to zero, it indicates that the shared memory is available to any requesting processor. Processors sharing the same memory segment agree to not use the memory segment unless the semaphore is 0, showing that memory is available. The processors also concur to set the semaphore to 1, while they are implementing a critical section, and then to clear it to 0 when they are done.

Testing and setting the semaphore is considered to be a critical function, and needs to be carried out as a single indivisible operation. Otherwise, two or more processors may check the semaphore simultaneously and set the semaphore in such a way that it can enter a critical section at the same time. This action allows the simultaneous execution of these critical sections resulting in incorrect initialization of control factors and a loss of necessary information.

A semaphore is initialized using a test and set instruction together with a hardware lock mechanism. A hardware lock is defined as a processor-generated signal that helps in preventing other processors from using the system bus as long as the signal is active. When the instruction is being executed, the test-and-set instruction tests and sets a semaphore and activates the lock mechanism. This helps in preventing the other processors from changing the semaphore between the time that the processor is testing it and the time that the processor is setting it. Consider that the semaphore is a bit in the least significant position of a memory word whose address is symbolized by SEM. Let the mnemonic TSL designate the “test and set while locked” function. The instruction TSL SEM is executed in two memory cycles, that is, the first one to read and the second to write without any interference as given below:

###### R  M[SEM] Test semaphore M[SEM]  1 Set semaphore

In order to test the semaphore, its value is transferred to a processor register R and then set to 1. The value of R indicates what to do next. If the processor identifies that R = 1, it means that the semaphore was initially set. Even if the register is set again, it does not change the value of the semaphore. This indicates that another processor is executing a critical section and therefore, the processor that checked the semaphore does not access the shared memory. The common memory or the shared resource that the semaphore represents is available when R = 0. In order to avoid other processors from accessing memory, the semaphore is set to 1. Now, it is possible for the processor to execute the critical section. To release the shared resource to other processors, the final instruction of the program must clear location SEM to zero.

It is crucial to note that the lock signal must be active at the time of execution of the test-and-set instruction. Once the semaphore is set, the lock signal does not have to be active. Therefore, the lock mechanism prevents other processors from accessing memory while the semaphore is being set. Once set, the semaphore itself will prevent other processors from accessing shared memory while one processor is implementing a critical section.

###### Notes



*Task* Visit <http://www.articlesbase.com/information-technology-articles/multiprocessor-> semaphore-318193.html and discuss how shared memory semaphores act as essential tools for interprocessor synchronization.

* 1. **Summary**
  + A multiprocessor generally refers to a single computer that has many processors.
  + The term ‘multiprocessor’ can also be used to describe several separate computers running together. It is also referred to as clustering.
  + Processors in the multiprocessor system communicate and cooperate at various levels of solving a particular problem.
  + The difference that exists between multicomputer systems and multiprocessors depends on the extent of resource sharing and cooperation in solving a problem.
  + Multiprocessor system uses a distributed approach, wherein a single processor does not perform a complete task but more than one processor is used to perform the subtasks.
  + There are two types of multiprocessor systems; they are tightly-coupled multiprocessor system and loosely-coupled multiprocessor system.
  + One of the major characteristics of shared memory multiprocessors is that all processors have equally direct access to one large memory address space.
  + Multiprocessor systems work efficiently as high-performance database servers, Internet servers, and network servers.
  + Common-bus multiprocessor system has numerous processors connected through a common path to a memory unit.
  + A multiport memory system uses separate buses between each CPU and each memory module.
  + There are many cross points located at intersections between processor buses and memory module paths in a crossbar switch organization.
  + The most frequently used procedure in a shared memory multiprocessor system is to set aside a part of memory that is available to all processors.
  + Appropriately functioning multiprocessor system must provide a mechanism that will ensure systematic access to shared memory and other shared resources.

## Keywords

***Autonomous Computers***: A network administered by a single set of management rules that are controlled by single person, group, or organization. Autonomous systems frequently use only one routing protocol even though it is possible to use multiple protocols.

***Control Logic****:* It is the part of a software architecture that helps in controlling what the program will do. This part of the program is also termed as controller.

***Multithreading****:* It is a process wherein the same job is broken logically and performed simultaneously and the output is combined at the end of processing.

***Real-time Applications****:* A real-time application is an application program that works within a given time frame that the user assumes as immediate or current.